
Submissions: This assignment is due in class on T Feb 18th 2020. Each student must submit his or her own assignment. Solutions can either be typed in Latex, MSWord or other such word processing software, or printed clearly. You must write your name and UUID clearly on your submitted assignment. For the programming problems the code must be in Python and should be submitted in a Dropbox folder (called “HW3 submission”).

Academic Integrity: You are encouraged to work in groups, but everyone must write out their own solutions. Absolutely no word to word copying is allowed. Please refer to the course policies and schedules about this. If you have worked with other students on the assignment or referred to external sources, please mention all names and sources on your assignment.

Partial solutions: If you are sure that you know how to arrive at a solution, but you get stuck in some place, it is better to write the partial solution. Honest attempts at partial solutions will be awarded.

4030/6030 points: This HW has no extra problems or a different points scheme for 6030 students.

<table>
<thead>
<tr>
<th>Name:</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID:</td>
</tr>
<tr>
<td>Email:</td>
</tr>
</tbody>
</table>
**Problem 1 [60 pts]** Write code in Python for the following problem: Given a list of numbers, any of which may be positive, negative or zero, write a divide-and-conquer algorithm to compute the maximum possible sum of numbers which occur as a contiguous subarray. If all numbers are negative for example, the output is the “empty” sum which is zero. If the input is,

10, 0, -1, 0, 0, 2, 2

the answer is 13. If the input is,

5, -6, 1, 1, 4

the answer is 6.

Also mention, as a comment in the code the runtime of your algorithm. For example, if your analysis comes out quadratic time you could say:

#The runtime is $O(n^2)$ where $n$ is the length of the list.

**Update 02/13/2020:** Please name your Python code file - `Maxsum.py` and the function, as `max_contiguous_sum(L)` where $L$ is the list argument.

**Problem 2 [40 pts]** Solve Problem 1.32 from the textbook. Remember that we analyze algorithms for their bit complexity and therefore analysis must be based on the number of bits taken to represent the number in question. An algorithm not polynomial in the input size will not be considered efficient.

**Hint:** Use binary search. We will cover this in class on Thursday (Feb 13).